DOCUMENT RESUME

ED 033 046 SE 007 570

By-Poore, Jesse H., Jr.; And Others

A Survey of Display Hardware and Software.

Georgia Inst. of Tech., Atlanta. School of Information Sciences.
Spons Agency-National Science Foundation, Washington, D.C.
Report No-GITIS-69-03
Pub Date 69
Note-48p.
EDRS Price MF-$0.25 HC-$2.50
Descriptors-*Computer Assisted Instruction, Computer Based Laboratories, *Computers, *Computer Science, Display Systems, *Information Science, Systems Development
Identifiers-National Science Foundation

Reported are two papers which deal with the fundamentals of display hardware and software in computer systems. The first report presents the basic principles of display hardware in terms of image generation from buffers presumed to be loaded and controlled by a digital computer. The concepts surrounding the electrostatic tube, the electromagnetic tube, and the charactron tube are discussed. Hardware characteristics of devices which allow interaction with a display system are also explained. The second report focuses on the software aspects of operating display devices. Programming considerations involving commands, interrupts, and pen tracking are discussed in a general setting. Data structures are then presented for expeditiously handling several geometrical forms that occur frequently in applications of computer graphics. (RP)

Work reported in this publication was performed at the School of Information Science, Georgia Institute of Technology. The primary support of this work came from the Georgia Institute of Technology and from agencies acknowledged in the report.

* * *

The graduate School of Information Science of the Georgia Institute of Technology offers comprehensive programs of education, research and service in the information, computer and systems sciences. As part of its research activities the School operates, under a grant from the National Science Foundation, an interdisciplinary science information research center. Correspondence concerning the programs and activities of the School may be addressed to Director, School of Information Science, Georgia Institute of Technology, Atlanta, Georgia 30332.

Telephone: (404) 873-4211

GITIS-69-03

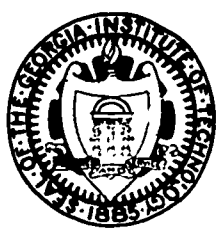
(Research Report)

A SURVEY OF DISPLAY HARDWARE AND SOFTWARE

Jesse H. Poore, Jr.

Jay E. Davidow

Donald P. Kelley



School of Information Science 1969

GEORGIA INSTITUTE OF TECHNOLOGY

Atlanta, Georgia

This report is dedicated to the
memory of our colleague and friend,
Jay E. Davidow, who died in an
accident on February 16, 1969.

ACKNOWLEDGEMENT

The work reported in the paper has been sponsored in part by the National Science Foundation Grant GN-655. This assistance is gratefully acknowledged.

VLADIMIR SLAMECKA
Project Director

ABSTRACT

The first report presents the fundamental principles of display hardware in terms of image generation from buffers presumed to be loaded and controlled by a digital computer. The concepts surrounding the electrostatic tube, the electromagnetic tube, and the charactron tube are discussed. Hardware characteristics of devices which allow interaction with a display system are also explained.

The second report is concerned with the software aspect of operating display devices. Programming considerations (e.g. commands, interrupts, pen tracking) are discussed in general form rather than for particular devices. Data structures are then presented for expeditiously handling several geometrical forms that occur frequently in applications of computer graphics.

CONTENTS

INTRODUCTION

In the very early stages of development of the now expanding Information Science Laboratory a wide variety of hardware was investigated to determine of each item its role, if any, in the conceptual laboratory. The class of objects for displaying data on a screen rather than on paper, was studied from our two essential points of view; as stand alone devices and as remote devices. Aside from the original purpose of the investigation of display hardware, a certain awareness of key considerations in display hardware developed in the graduate assistants doing the work. It was therefore decided to write these impressions and their generalizations for the laboratory files and for the benefit of students who might later examine the state of the art of display hardware.

The first paper was prepared from a basic design point of view and develops systematically to the software writer's level of interest and on, briefly, to the user's. The second paper was motivated by experience in large systems development and is primarily concerned with data structures. Any overlap between the two papers must be accepted as simply points of contact between the two approaches.

As one would expect, the principles of display hardware are quite explicitly stated in engineering terms. The task is to distill from the technical writings those details that are truly fundamental and to present them in such a way that a nonelectronics type (though not an absolute layman) can understand them. In this sense then the paper on hardware principles is quite successful and should retain its validity and usefulness for some time despite periphery developments and changes in the industry. The main stream of the paper is to take a simple abstract display system through several levels of complication in terms of functional components. While specific devices are handled as asides and illustrations, these do not obstruct the overriding objective of the paper to elicit the fundamentals of display hardware.

In contrast to the hardware situation, the principles of display software are very evasive. (Display software is not unique in this respect.) This is not to suggest a scarcity of literature devoted to the topic of display software, but to point up the ad hoc nature in which software emerges. Nevertheless, the broad categories of software considerations are identified and briefly discussed. List structures and subpicture structures are identified and the reader is alerted to their prominence in the literature. A bibliography is supplied and it should be observed that this bibliography is rapidly becoming history. If there are, indeed, principles of display data structures they have not as yet been identified in the literature and so are not presented here. We do have here, however, a very good feel for state of development of (display) data structures and a clear suggestion that this area come under intense research.

J. H. Poore, Jr.

PRINCIPLES OF DISPLAY HARDWARE TECHNOLOGY

Jay E. Davidow

Research Associate

In the past several years display hardware has become commercially available and its use is rapidly increasing. The advent of integrated circuit technology, experience in the development of cathode ray tubes, and particularly overall improvement in the performance of computing systems have all contributed significantly to this end. A great factor in the push to this development has been the so-called "information explosion." Within a few years after the widespread acceptance of data processing devices, it became obvious that output production was severly limited as to: form (maps could not be produced for the military), and speed (searching files for data was slow). Further useless reams of paper were proliferated simply to provide limited visualization of data (for instance, the deluge of teletype output of computer programming activity at time-sharing terminals). Though displays are not a panacea for these problems, they provide a definite improvement over some methods.

Basically, a computer-driven display device consists of a cathode-ray tube used for optical presentation, the associated amplifiers, and the interface to the computer. In this paper, esoteric generation techniques, e.g., the laser-holographic displays, will not be considered. Instead, an understanding of the operation of general type of CRT display will be presented. More sophisticated displays, such as film displays, will not be mentioned, but it is felt the reader will be able to fathom the operation of these at his leisure. Instead, emphasis will be placed upon structure and theory of general display types, and application of some of the principles presented within will be apparent, with reference to other types.

The computer interface is the differentiating factor in the sophistication of most graphical displays. This interface may be as simple as control lines led to the amplifiers proper, and may evolve to be small, special-purpose computers in themselves. A simple display interface is shown in Figure 1

The Command decoder interprets the control signals arriving from the central processor to perform the desired function. For example, let the

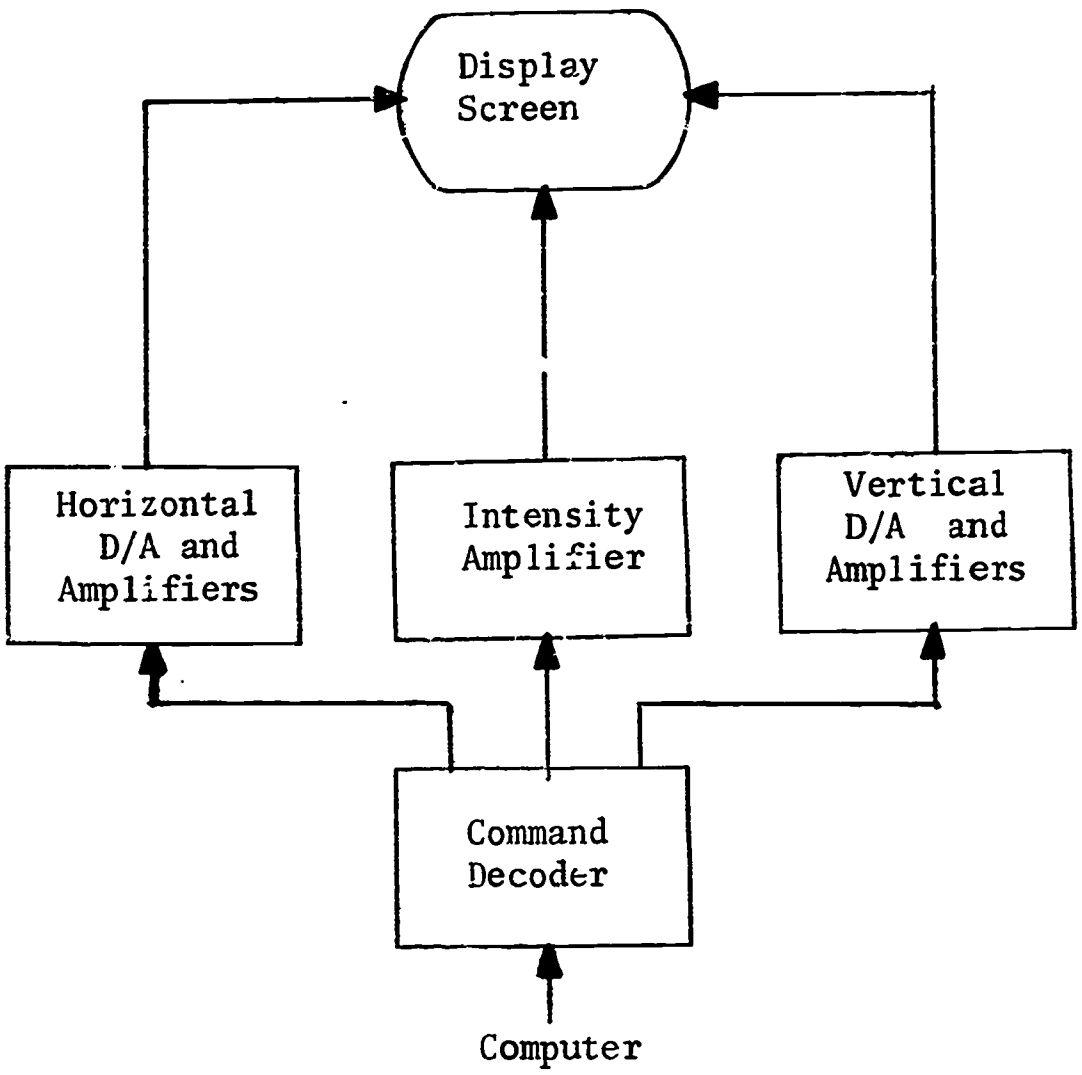


Figure 1: Simple Interface

simple display have two programmed operations:

- display a point
- display a vector from current position.

To cause a point to be displayed, the programmer issues an I/O instruction, which directly causes the command logic to force display of the desired point at (x,y). Of course, the coordinates must be available: this may be either from core memory, on active register in the central processor, or a pre-loaded buffer in the display. To the command repertoire, let us add:

- load x buffer register (in the decoder) from A_1 (core memory)
- load y buffer register from A_1.

The sequence of operations to display the point at (x,y) would now be:

- load x buffer
- load y buffer
- display point at (x,y).

On another machine, it may be possible to specify both the desired operation ("display a point") and the coordinates in a simple operation ("display a point at (x,y)").

Cathode-Ray Tubes

Certain phosphors emit sharp light when hit by an electron beam. If a device can "capture" an electron beam and place it in a desired location, a rudimentary CRT display results. There are two methods of "capturing" and "positioning": electrostate and electromagnetic. In the electrostatic tube, a static electric charge is employed to control the motion of the beam. Since

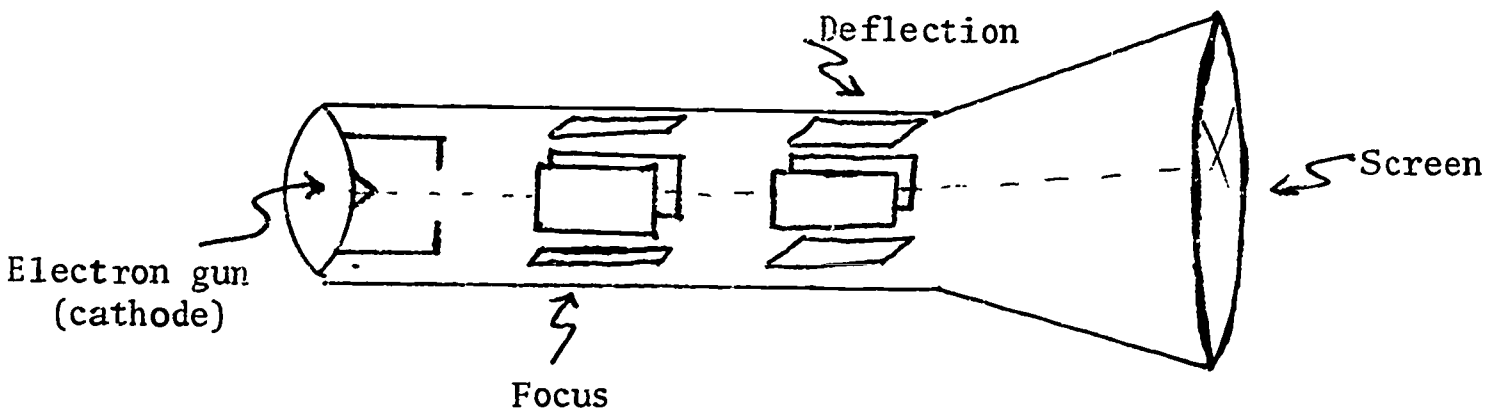


Figure 2: Electrostatic Tube

normally a discrete, precise point of light is desired, the electron beam is "shaped" to a fine point by focusing plates. This, in the electrostatic case, is the result of placing high voltage electron charges on several plates surrounding the beam as it leaves the gun. Traveling at high velocity, the beam then passes through two sets of deflection plates, which

cause the beam to be shifted, either horizontally or vertically, to the desired target on the screen. These deflection plates are also electron charged surfaces. Since the beam consists of electrons, and the plates have unequal electron charges, motion toward the plate of lower charge (with respect to the voltage of the cathode) is achieved. Electrostatic deflected cathode ray tubes have the advantages of being highly responsive to changes in deflection, lightweight, and relatively uncomplicated in amplification. This is offset by losses in the focusing of the beam near the edge of the display field, a large size required for the tube, and a relatively complicated structure.

Electromagnetic cathode-ray tubes are an improvement over electrostatic tubes if optical qualities are critical. Beam quality is consistent over the entire display area, the beam is very sensitive to deflection charges, and the display is very intense due to sharper focusing. However, this is gained at an increase in amplifier complexity and weight and a loss in beam speeds. Basically, an electromagnetic tube is similar in concept to the electrostatic tube, with magnetic fields, instead of electric fields, producing the deflection and focus. In the former case (EM), all the mechanisms, except for the electron gun (cathode) are on the surface of the tube, thus decreasing the complexity of the tube itself. Magnetic coils outside the tube replace the electron plates inside the tube; these coils usually rest on the outer "skin" of the tube.

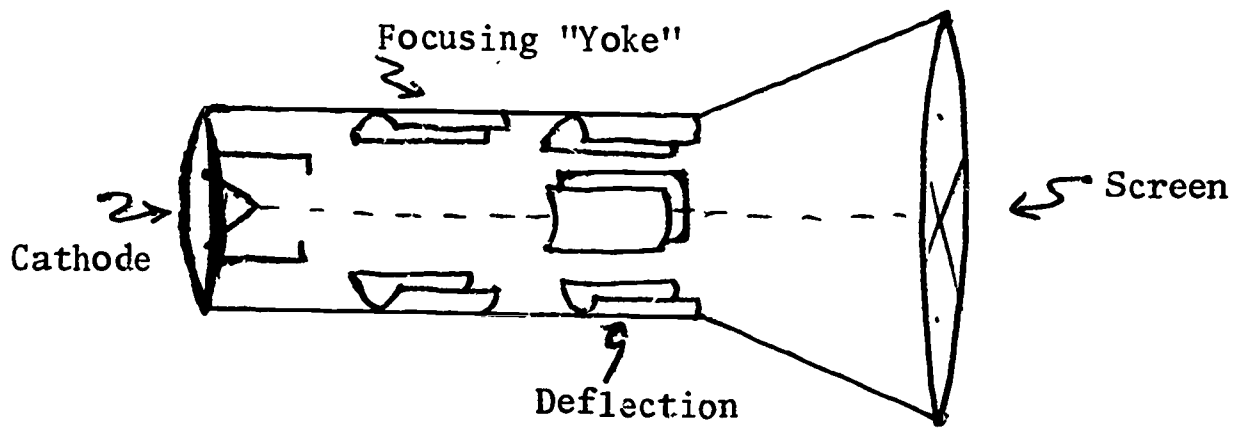


Figure 3: Electromagnetic Tube

For both cases, however, the logical structure of the display is the same. The interface (or command decoder) specifies that the digital coordinates be converted (in a digital-to-analog converter: D/A) to voltages or currents and then amplified. This causes repositioning of the beam; the intensity is either turned-off ("blanked") during repositioning for the display of a point, or turned up ("unblanked") to generate a vector. Control of the intensity is accomplished by the command decoder.

Human Factors

The average human eye is capable of distinguishing (or resolving) two points on 1 minute of an arc at 18". Geometrically, then, two lines .005" apart would appear distinct, but most CRT's are capable of focusing the beam spot size to only between .010" and .015". This rarely causes conflict, except in highly complicated graphics. Secondly, the eyes respond to changes in the visual image at a maximum of about 20 to 25 times a second (40 to 50 milliseconds). At any rate slower than this, images appear discrete; greater, images appear continuous. Hence, American television systems scan at 30 frames a second. This is significant when considering the "flicker" problem, flicker being the condition of displays when the eyes perceive this separate image quality. Older home movies often exhibit this. Much work has been done studying the problem of flicker. One interesting solution is to generate the display in a seemingly random pattern, that is, instead of producing the image from right-to-left or vice versa, randomly generate portions of the image. This uses the inhibition qualities of visual perception; once excited, eye cells do not respond to another excitation for 40 to 50 milliseconds. Thus by overlapping images (chronologically) from different portions of the screen, the critical flicker rate is dramatically reduced.

Character Generation Systems

By far, the most general use of displays is for text display. Often, this can be accomplished by generating characters composed of individually

displayed points or vectors. This has the inherent disadvantage of being time-consuming and somewhat cumbersome, although it allows for great flexibility is the generation of character sets. In systems where time is of a premium and/or much of the work is of a character nature (fonts not being significant), the control logic or CRT contains a character generator. Several different types of character generators are now being employed.

A novel type of character generator is that used by the Stromberg-Carlson Company in its CHARACTRON tube. Inside of the tube is a matrix filled with character shaped openings; the beam is directed to a point on this matrix and redirected to the screen.

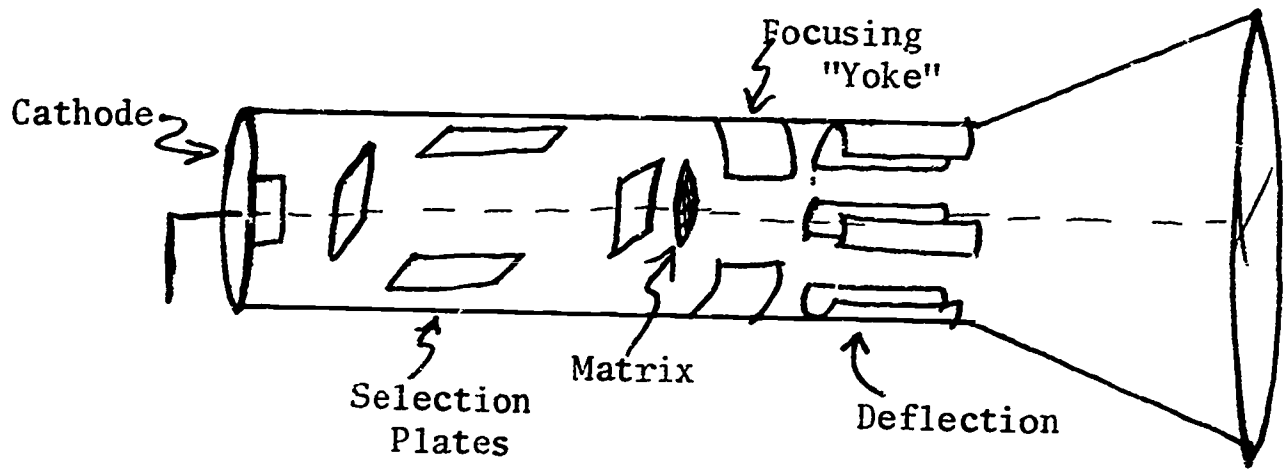


Figure 4: Charactron Tube

At manufacturing time, a matrix of selected characters is assembled into the tube. By using a dot on the matrix (e.g., a period or point), conventional points and vectors may be exhibited. One disadvantage to this scheme is the immense complexity of the tube and its driving mechanisms. In the past, there have also been questions as to its resolution capabilities and speed. However, at the 1st National Symposium on Information Display in

1963, Stromberg-Carlson reported that random display rates of 50,000 characters per second were available then. The logic required to use a CHARACTRON is a great increase over the simple system previously described. An additional instruction to cause character generation would have the command decoder produce the required signals to be applied to the selection plates. Indeed, display of a point might be just the same as display of any character, if so desired.

Other types of character generation are associated with electronic digital logic or external amplifiers attached directly to the horizontal and vertical deflection mechanisms, as below:

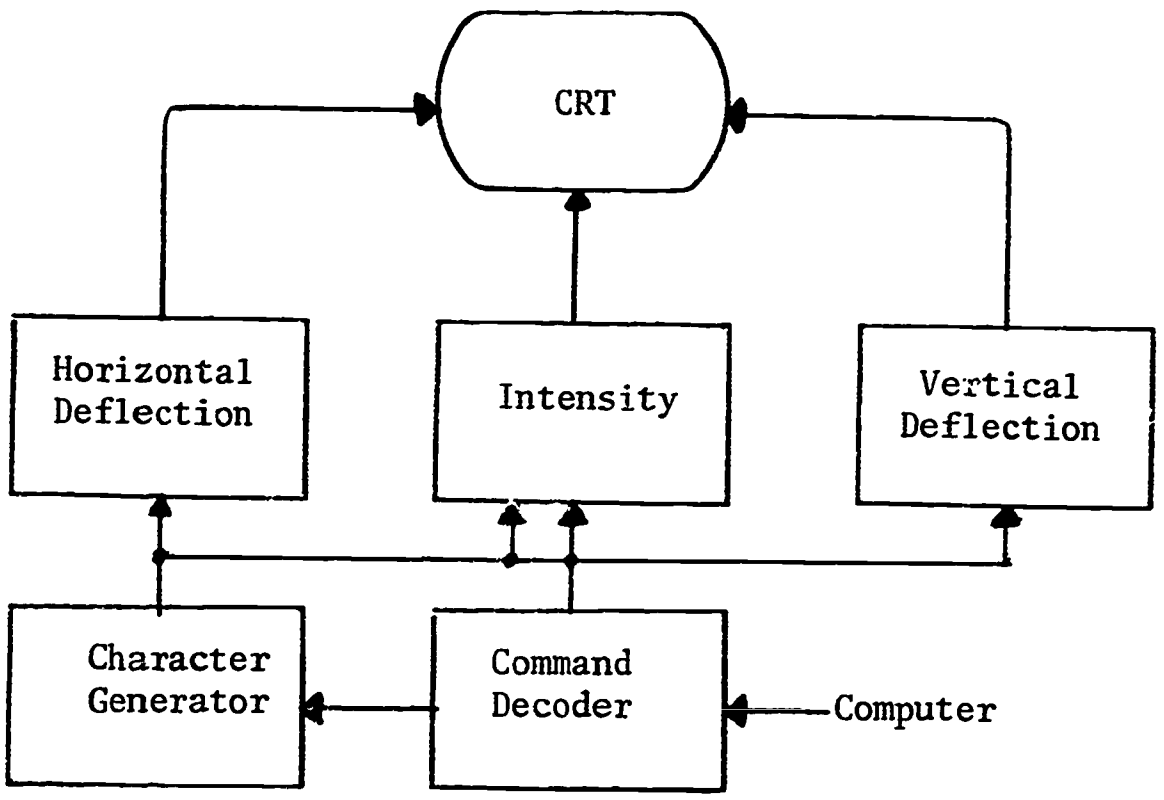


Figure 5: Character Generation

There are four major types of character generation, dot patterns, stroke patterns, Lissajous patterns, and scan patterns. Since each type of generation pattern has qualities that are worth discussing, but

invisible to the programmer, before going any further the simple display may now have the following instructions:

- display character at (x,y)
- load character generator buffer.

a) Dot patterns

In this procedure, a character is represented in the generator by a series of analog steps. Conceptually, it is very similar to programmed character operation; the hardware now provides the pattern to be traced. At the desired (x,y) coordinates, the character generator begins to step its way through a logical matrix on the CRT face. At selected points, the beam is unblanked; this produces an image of the character in the form of a dot sequence:

```
7    . . .
6   .     .
5   .     .
4   . . . . .
3   .     .
2   .     .
1   .     .

    1 2 3 4 5
```

In the 5x7 matrix form, the x deflection may be incremented 5 times while the y-deflection may be incremented 7 times for each step of x. Thus, the δx to the x:D/A converter may appear as in Figure 6.

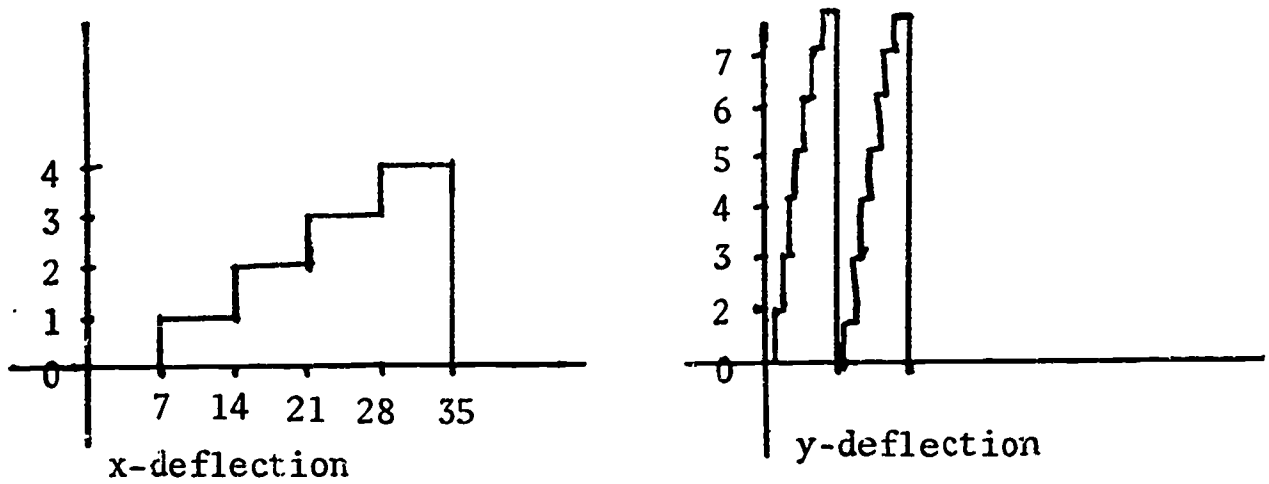


Figure 6: δx to x:D/A Converter

The beam is then unblanked as in Figure 7.

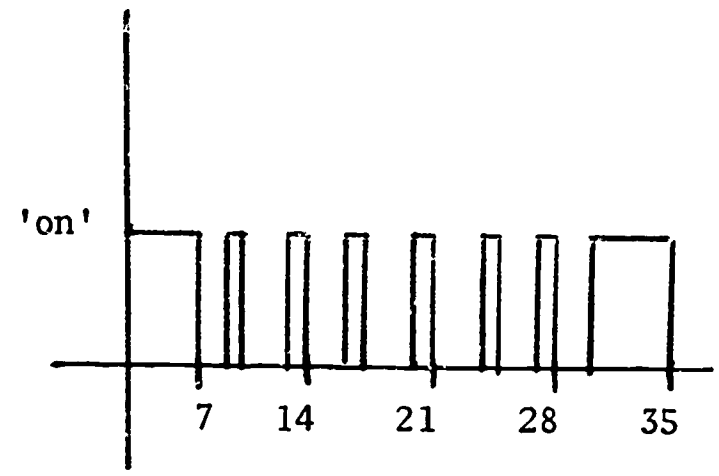


Figure 7: Intensity

Sometimes, rather than use a synchronous method requiring a fixed number of cycles, the character logic is such that only the unblanked points in the matrix are addressed:

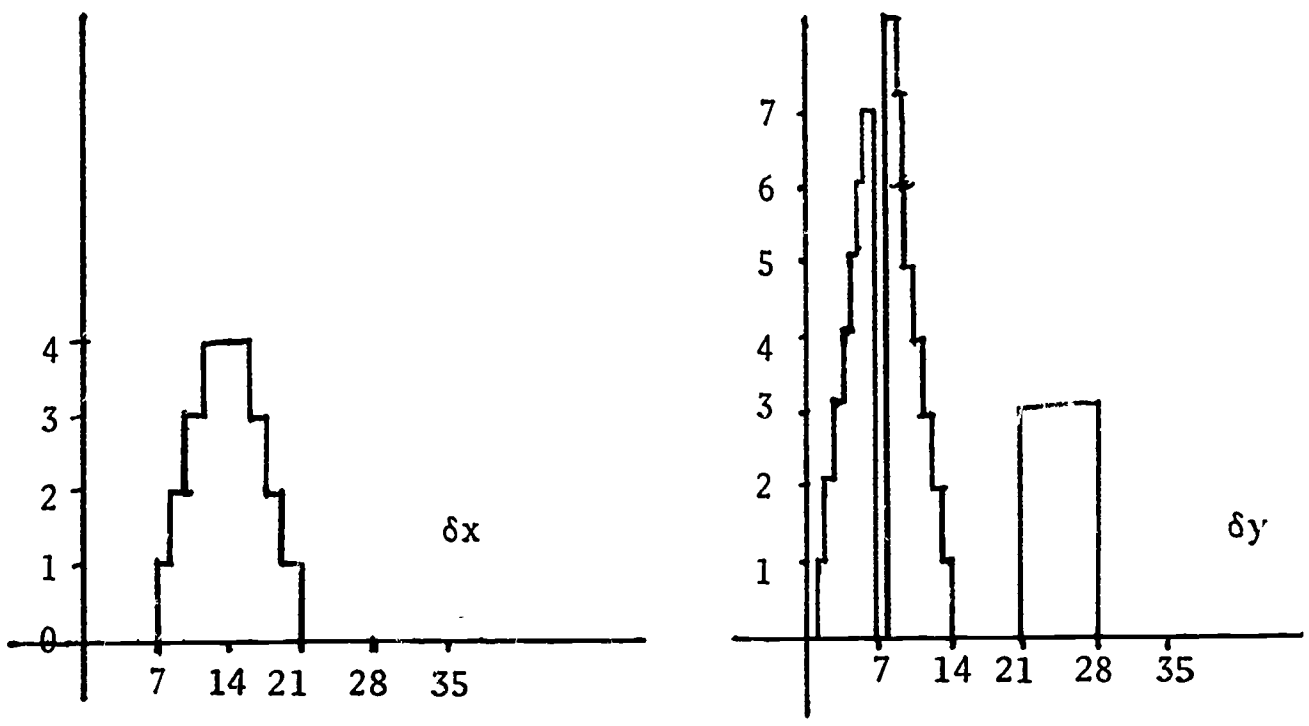


Figure 8: Variable Dot Format

This allows for a variable format dot matrix, but results in complications in timing and addressing control of the screen.

b) Stroke patterns

With this algorithm, characters are produced as the result of a series of strokes generated in a rectangle. As with the dot pattern, character forms are stored either in logical form or analog signals. The bottom left of the rectangle is the (x,y) coordinate; usually, only the pertinent unblanked strokes are traced, but it is possible to trace all strokes by adding an intensity control. The strokes appear thusly:

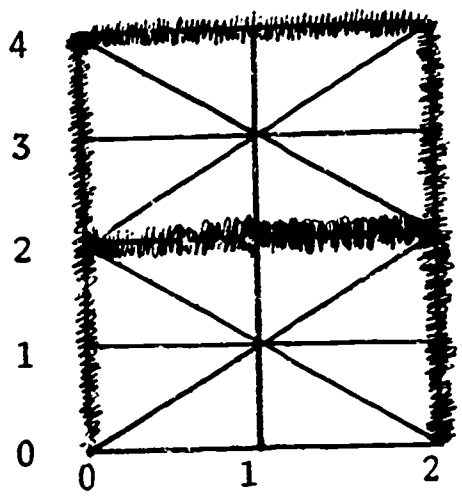


Figure 9: Stroke Generation

Assume all strokes are to be unblanked, then the x:D/A and y:D/A deflections are:

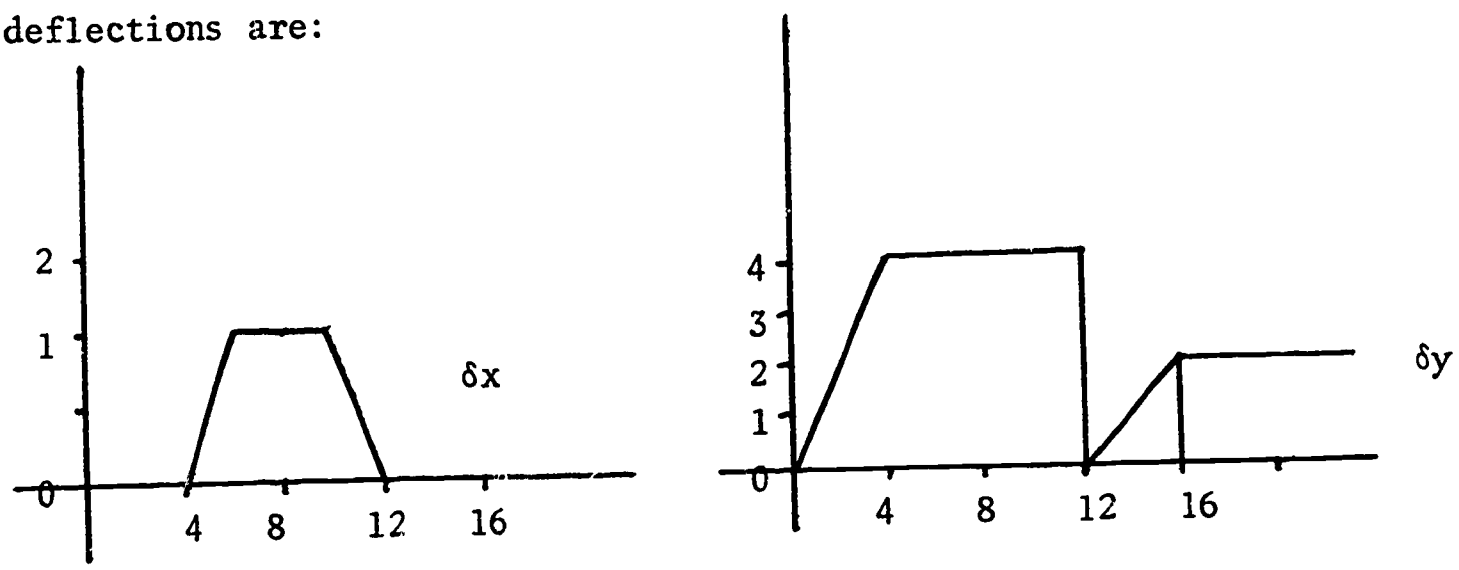


Figure 10: x:D/A and Y:D/A

Since retracing is necessary to "cross" the A, the intensity is blanked for a portion:

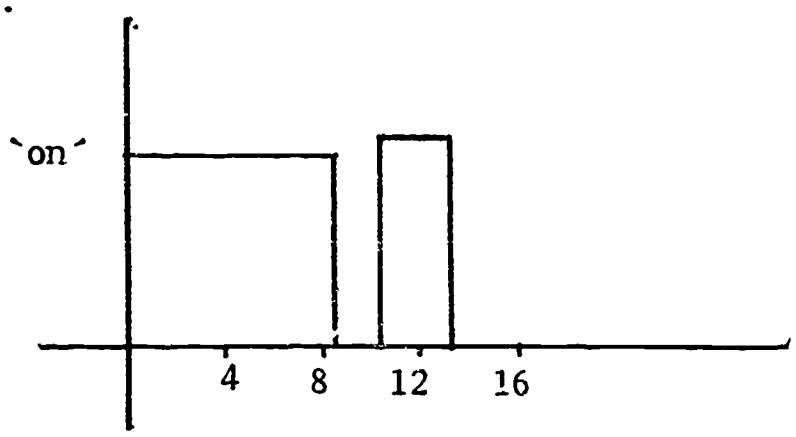


Figure 11: Intensity

The diagonals are obtained by combining the four basic strokes, e.g., to obtain:

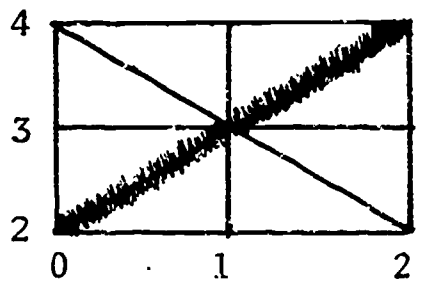


Figure 12: Diagonals

The following x:D/A and y:D/A deflections may be used (from time zero):

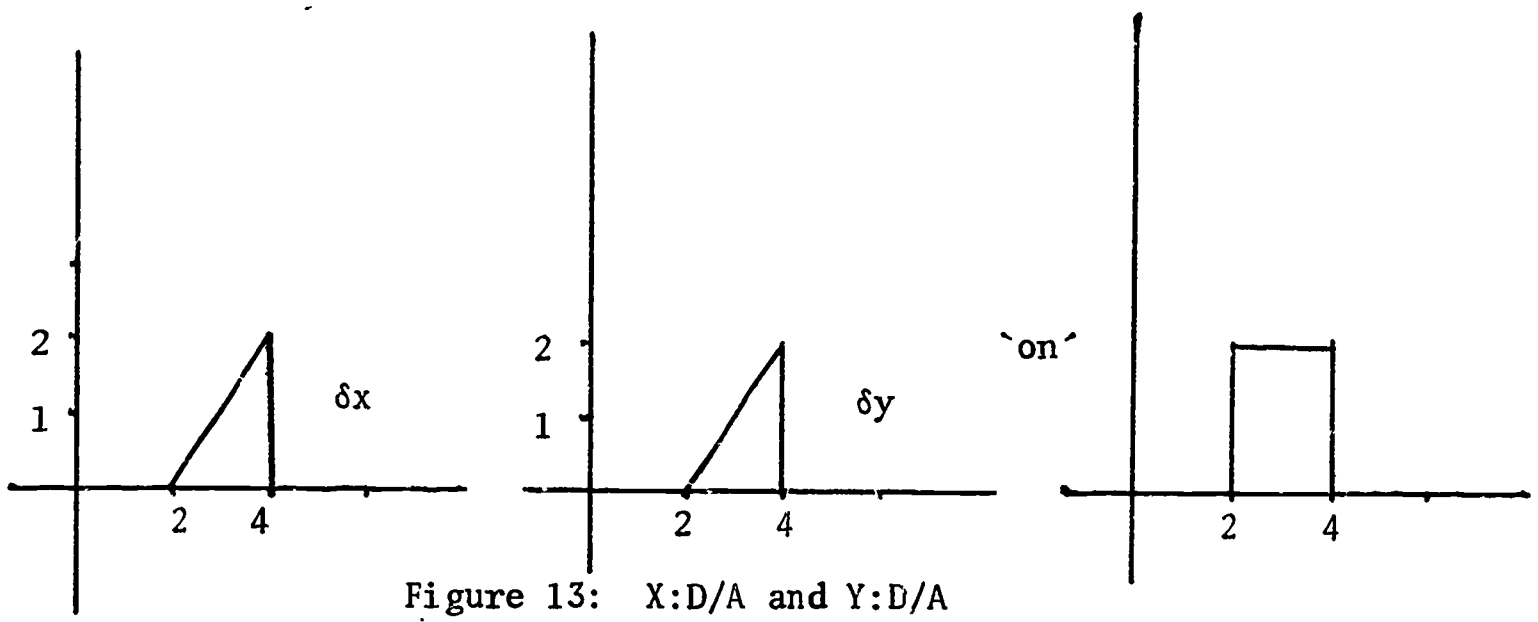
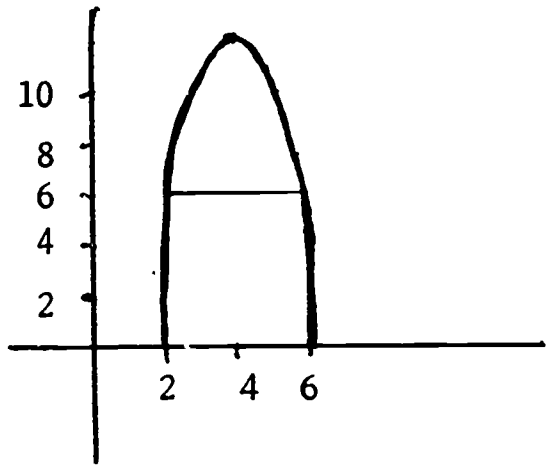


Figure 13: X:D/A and Y:D/A

c) Lissajous Patterns

A Lissajous pattern is the result of plotting the sine function on the abscissa and another sine of integer multiple period on the ordinate of a Cartesian plot. If the two periods are equal, a circle results; if they are not integer multiple, an elliptical shape ensues. The most common method of using these patterns is to combine the wave-forms with other periodic functions such as sawtooths and square waves to produce the letter A, as displayed below



The wave-forms might be shaped as follows:

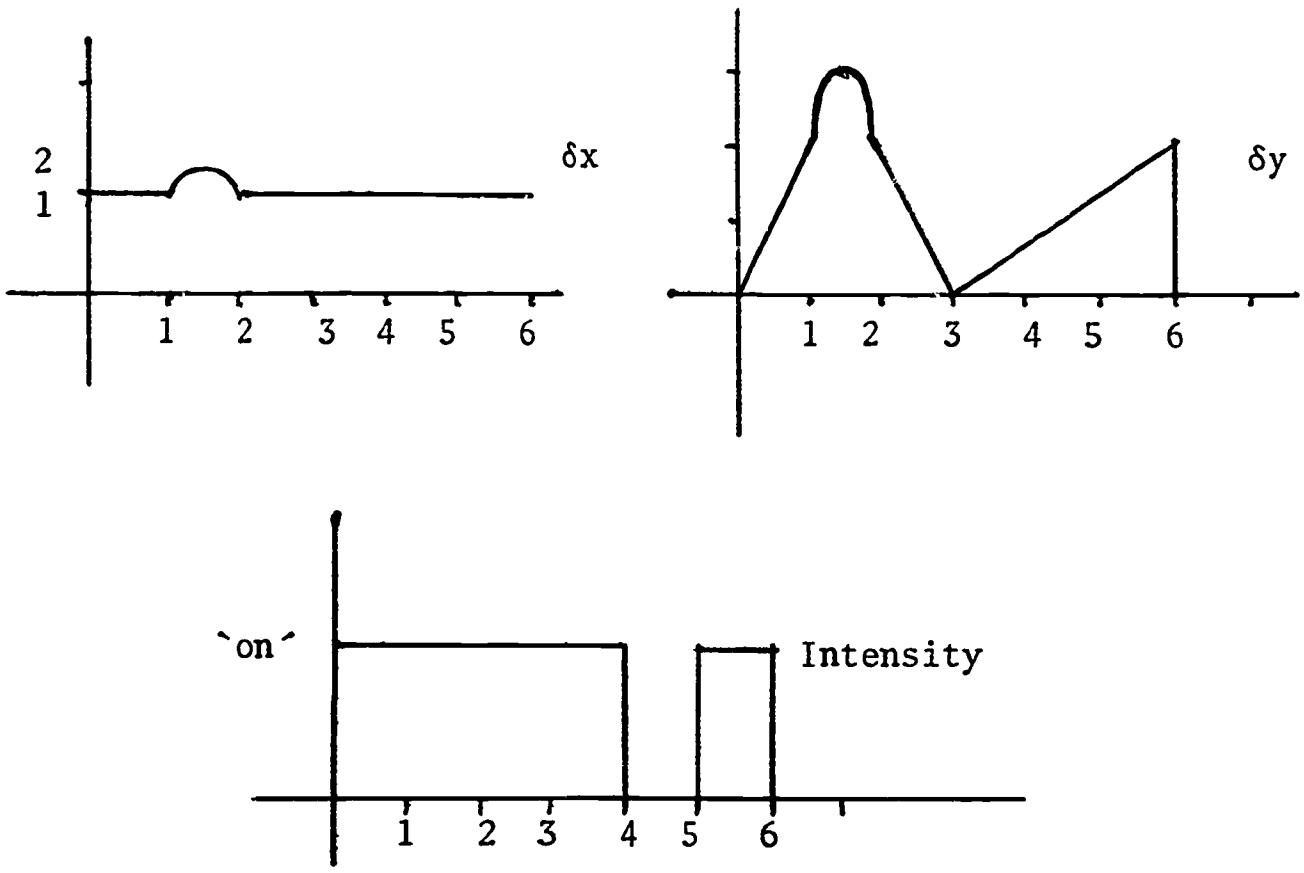


Figure 14: Production of "A"

With this method, better shaped letters are produced, but at an increase in cost and time. The matrix of characters is a set of wave-form selectors in periodic order along with intensity control. The drawings above are not to scale since scaling would complicate the principles of operation. The generator consists of wave-form generators and mixers which are applied to the deflection plates. Unlike the two preceding discussions, this method is purely analog.

d) Scan Pattern

This method is most commonly used in scan-type display systems, such as television displays. Letters are generated by a series of horizontal sweeps with blanking control. Empirically, a minimum of nine sweeps is required for acceptable generation. Though digitally addressed display systems rarely use this method, conceptually, it operates by always sweeping the same area with line segments, while the generation storage contains the intensity modulation information. To produce an A, for example,

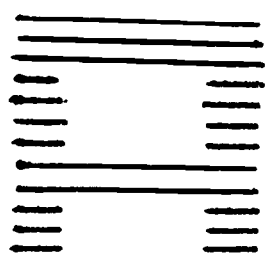


Figure 15: Scan-Pattern "A"

The deflection might be stepped up per Figure 16.

Command Decoder and Interface

To implement the character generator, two commands have been added to the display instruction repertoire:

- display a character at (x,y)
- load character buffer

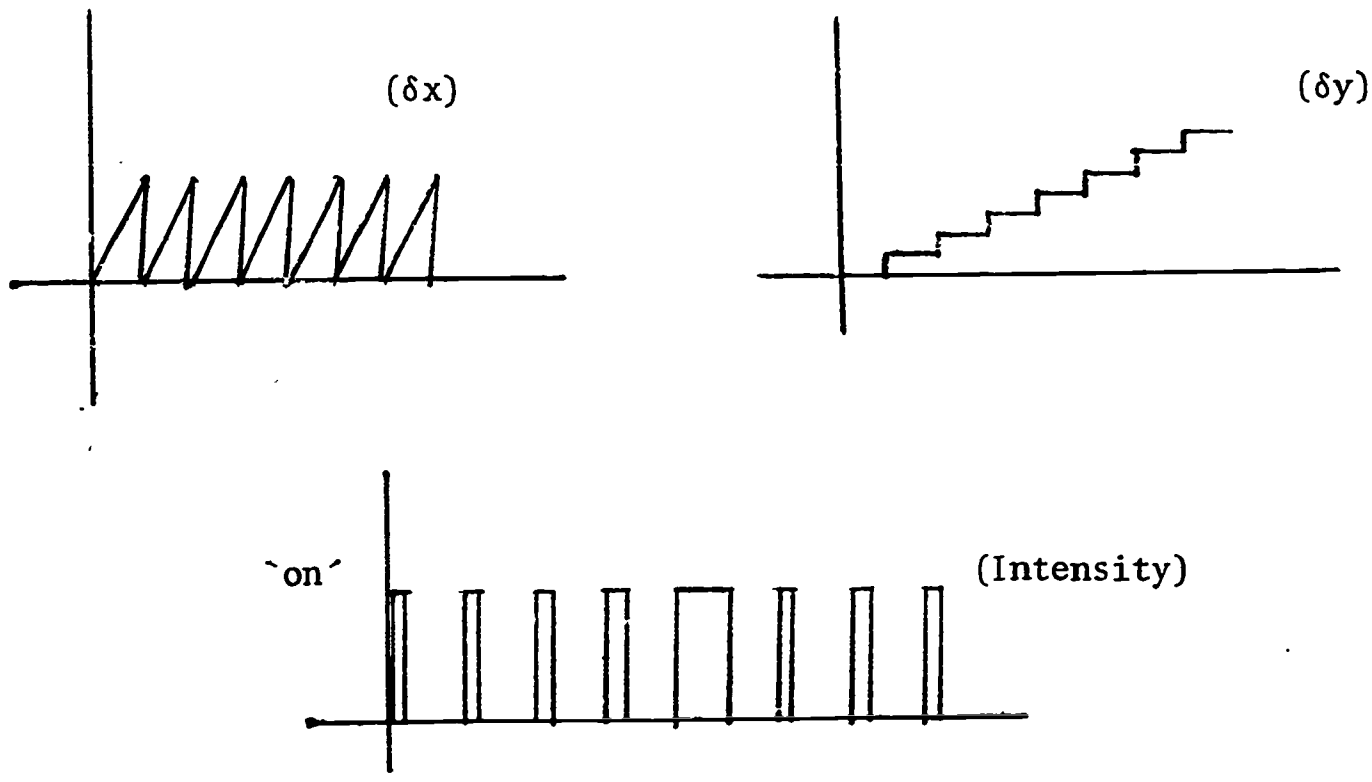


Figure 16: Deflection

plus the other instructions:

- load x buffer
- load y buffer
- display a point at (x,y)
- display a vector from current position to (x,y). The last instruction merely causes the beam to be shifted from its current position to the new position with the beam unblanked. To display a point, the blanked beam is shifted and then unblanked at the desired position. Many displays, though, are constrained to limit the angle of deflection on the tube face. To avoid this problem, some displays have line generators, which contain the necessary logic to blank and unblank the beam at the current position. With the line generator included, the block diagram expands.

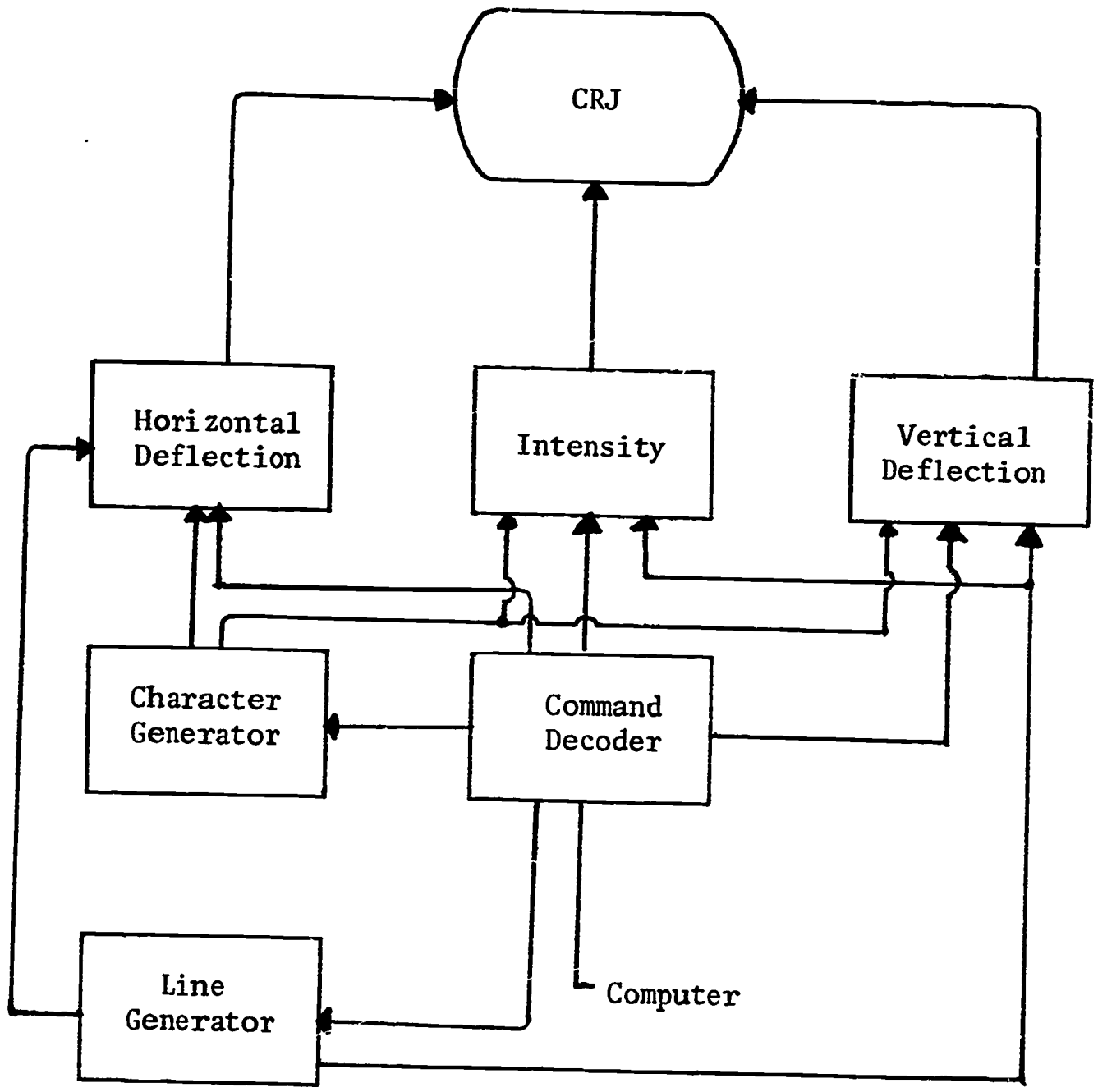


Figure 17: Expanded Interface with Line Generator

However, the simple display still has several undesirable features:

- the display image must be regenerated by the driving computer;
- moreover, the display is an output only device.

The former problem is of great concern particularly when the image is

largely graphical and the result of large computations, or when several displays are attached to the same computer. Regeneration, of course, is necessary to produce a new image or to keep the present image on the face - the latter usually referred to as "refreshing." If the image is not refreshed enough, flicker results. For the purposes of slowly changing displays, storage tubes are often employed. In this device, the cathode-ray tube screen has a highly persistent phosphor which loses the image very slowly. There are problems which arise from the use of storage tubes -- often, some resolution is lost -- which are dependent upon use, particularly if light pens are used. (Light pens will be explored later.)

Another more expensive solution is to provide a "buffer" between the computer and the command decoder. Buffers are of varying sizes, dependent upon complexity of the use. Operation is fairly straightforward. The computer loads the buffer and initiates action on the display. The command decoder has additional logic to read the buffer and obtain its instructions. This method reduces interaction between the computer and the display, but at the expense of more sophisticated programming. Refreshing is usually accomplished by a "transfer in buffer" command back to the beginning of the display image. With the addition of a decoder, the display unit approaches the level of a small control unit or special purpose computer. There are now two sets of instructions to the display: the previous set dealing with image generation and a new set dealing with control of the buffer and command decoder:

- load buffer
- begin execution of display commands at location A_1
- halt execution of display.

To further simplify operation of the buffer, often two sets of image instructions are used: one set to put the device in the various modes, character, line and point, which will be followed in the buffer by data, and the other to control image operations. Thus, the sequence of operations to generate a triangle filled with points and labeled "TRIANGLE" might well be:

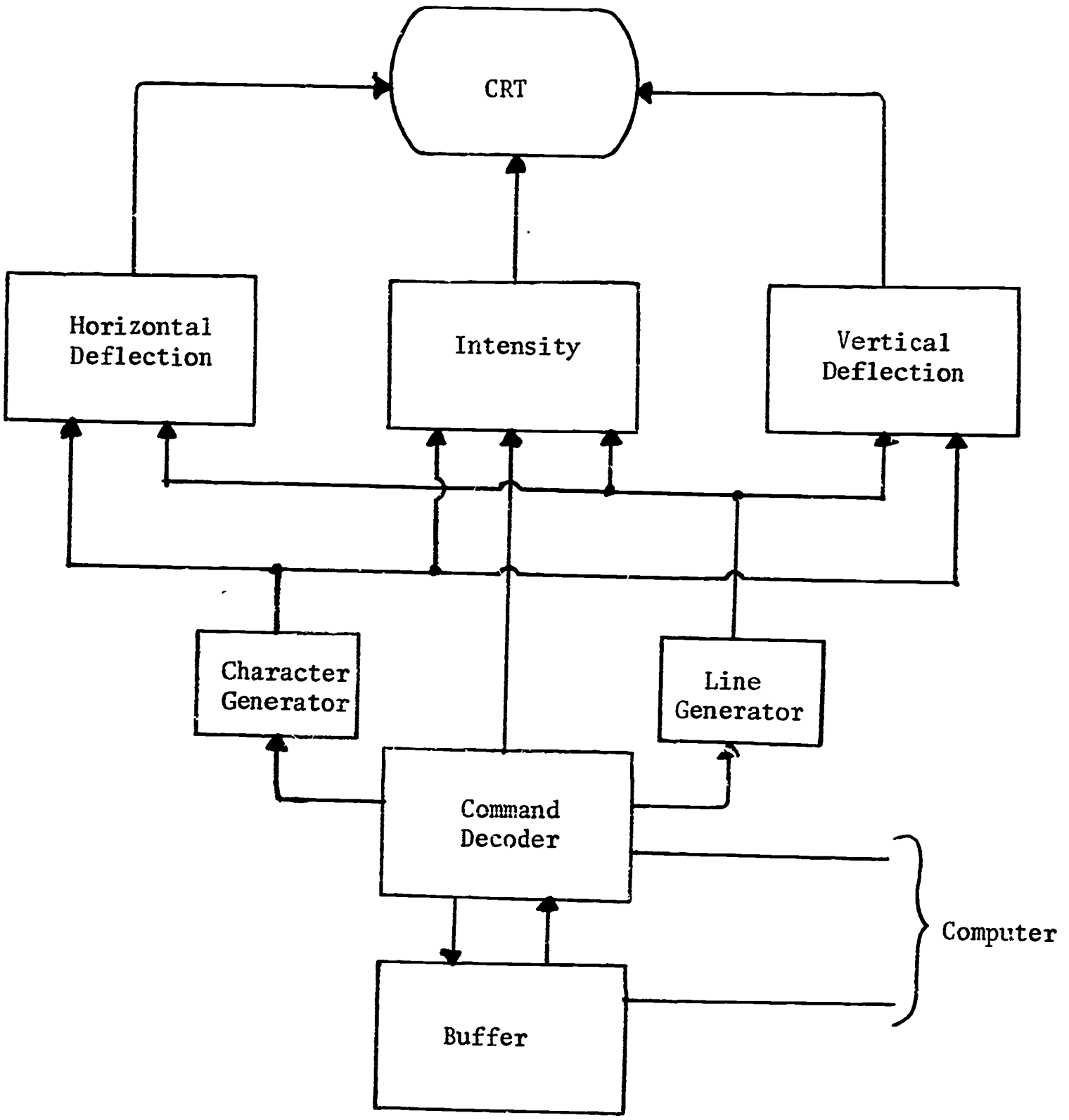


Figure 18: Interface with Decoder

a) - put display in time mode

$(x_1\ y_1)$

$(x_2\ y_2)$

$(x_3\ y_3)$

$(x_1\ y_1)$

b) - put display in point mode

$(x_1 + 1,\ y_1 - 1)$

$(x_1 + 2,\ y_1 - 2)$

$(x_1 + 2,\ y_1 - 3)$

.

.

.

$(x_3 + 1,\ y_3 - 3)$

c) - put display in character mode

T R I A N G L E

d) - jump in buffer to a).

To load the buffer and begin execution,

- load buffer

(assume this is a block transfer of the whole buffer)

- begin execution of buffer at a), and the display will continuously have the image depicted in Figure 19.

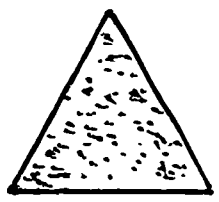


Figure 19

In some cases, the command decoder may not have the ability to set its mode from the buffer. Instead, the driving computer may issue a control operation to put the display in line mode, and then fill the buffer with only the desired points. Another control instruction (from the computer) is then issued to put the display in point mode, and then the buffer is loaded with point coordinates, and so on. In this case, refreshing still has to be initiated from the driving computer, but it is not dedicated completely since the buffer is loaded via block transfer. However, a worse case results when data cannot be ordered to only points, then lines, then characters, and more processor time is spent per image:

- load buffer from A_1
- begin execution
- place in line mode
- load buffer from A_2
- begin execution
- etc.

Many displays have only point or only character capabilities. This decreases the expense of the display, since fewer operations are required (no mode settings, for example). The most common of these are character-only scopes used in commercial operation (the term scope equivalent to display).

Interactive Devices

Thus far, the display system has been a simplex device, capable of producing graphical output solely. Since the display must operate, by nature, in an on-line environment (with possible exceptions such as storage tubes), some form of communication to the computer is desired. A display system usually provides some device with which the operator may affect system operation.

Light Pens

Light pens are light sensitive devices which use the generated luminescence of the display face to relay information to the driving system. A photoelectric pickup is located in the pen housing or the light is directed via fiber optics to a photoelectric pickup and when sufficient excitation (usually a single point on the screen will suffice) of the photocell occurs, registration of this fact is either transmitted back to the computer or analyzed by the display controller. The x-coordinate and y-coordinate buffers are then sampled to determine the location of the detected point. This may either be done by hardware logic in the controller or under program control, the former is usually the case in "tracking" pen systems.

A tracking pen system is one in which the pen may be used to trace a shape on the display screen. Often, a tracking cross is employed; the pen detects one edge of the cross and transposes the cross to center it at the detected point.

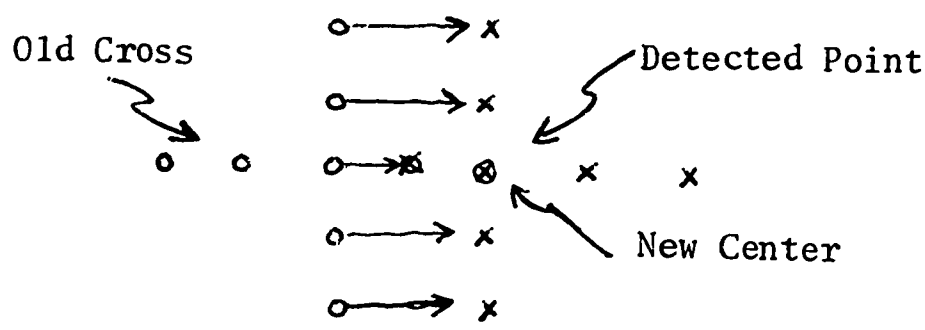


Figure 20: Tracking Cross

Since the radius of detection of the pen is usually greater than the mean radius between points, the tracking cross is fixed to move only laterally or vertically. To provide for tracking at angles, several tracking algorithms have been developed. These, in general, shift the cross to meet the detected point at successive approximations. One example is the spiral track.

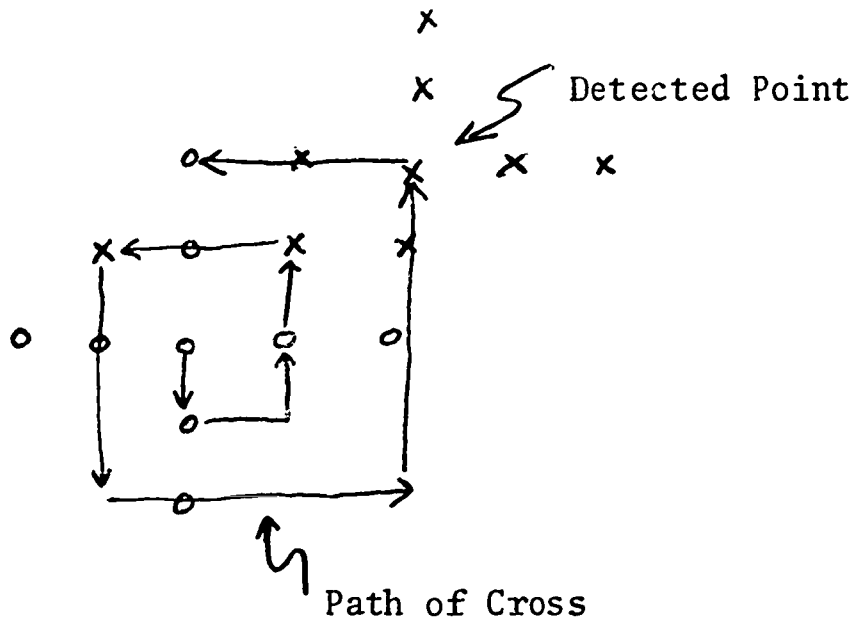


Figure 21: Spiral Track

However, the hardware required for pen tracking is very complicated and expensive, since displacement generators and comparators are necessary. If the pen is used, but not needed to track new lines or points, a "pointing" pen is employed. With this method, a tracking cross is not generated, and only points currently being displayed (by program control) are detected. If an isolated point is necessary, then the screen is flooded with points for a single cycle, not disturbing the image greatly. Then the x- and y- registers are read for the coordinates of the detected points.

Assuming that the light pen has some manner of informing the central processor upon detection of a point, via the interrupt mechanism or some flag that is set, it is possible to add the following instructions to the display system repertoire:

- read x coordinate to A_1
- read y coordinate to A_2.

Note that the existence of a tracking mechanism is independent of this model, since the detection of a point must be registered in either case.

A typical sequence of instructions to interact with the light pen might be:

- load display buffer from A
- start buffer execution at B

- transfer to l_1 at pen flag off
- read x coordinate to A_1
- read y coordinate to A_2

⋮

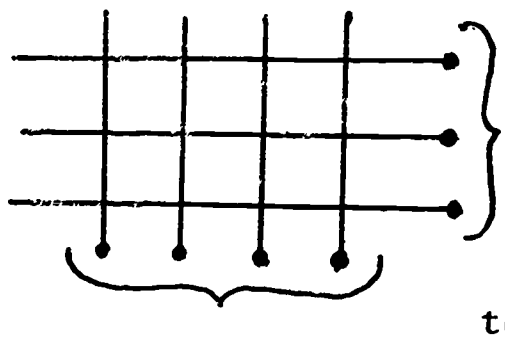
l_1 - ...

The program following the reading of the coordinate buffers probably would perform some comparisons to find at what image element the detect occurred or to start the generation of a new element at the detect position.

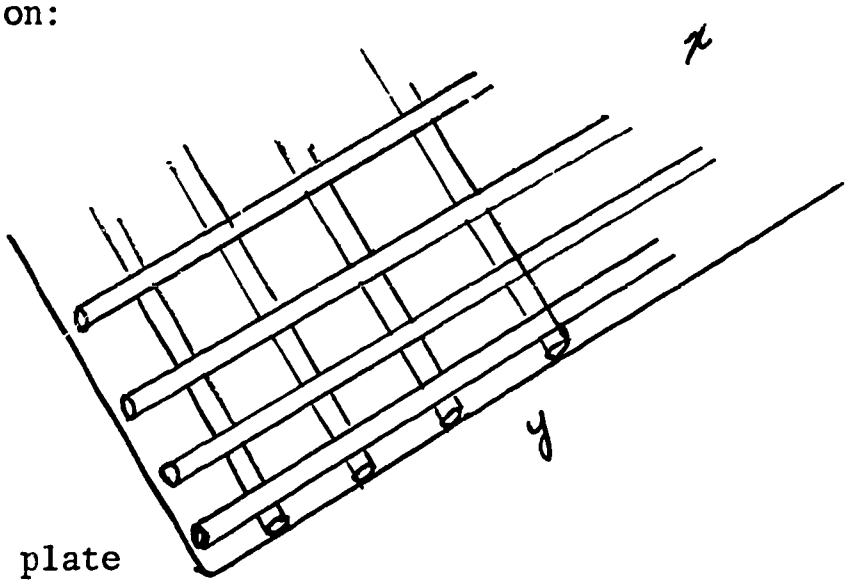
Writing Tablets

A novel approach to the solution of the interaction problem has been the introduction of such devices as the Rand or Teager tablet. These devices avoid the complications in hardware of light pens since they operate independently of the display generating hardware. In fact, (for the most part) the only link is a logical connection in the program. Generally, the tablet is a representation of the screen face, and is addressable analogously (of course, input-addressed only).

When pressure is applied to a point on the tablet, an address is generated by microscopic wires layed in perpendicular directions within the tablet. As a simple model, assume that the tablet appears, from the top:



and, from a cross-section:



electrified plate

The layers of wire are separated, for example, by a dielectric. As pressure is exerted, the dielectric is distended and the capacity changes. This causes an increase in potential on particular x- and y- lines and this change is sensed. For simplicity, assume that the threshold of this difference is fixed and a maximum to be selected can occur at only one point. These threshold decorders in turn drive logic which generates the x- and y- coordinates of the detection. In reality, tablets are not as unsophisticated; some generate digital addresses directly and others generate analog frequencies corresponding to the location. However, it is not necessary to cover these actions in detail.

Note that this action is completely independent of image and is implicitly capable of tracking (if the device will respond quickly enough). This is a significant improvement over light pens, but at an expense. The operator no longer can select an image directly from the screen; to provide for coordination, a pseudo-tracking cross must be established. After the image has been selected, then the addresses are compared (as with a light pen) to the display sequence. From this juncture on, the two devices are handled identically. Indeed, depending upon the system, the light pen and tablet are logically identical or near to that.

A novel modification of the tablet is that used by the Illinois Institute of Technology Research Institute (IITRI) on one of their display systems. In this system, the stylus is suspended over the writing surface

by diagonal lines connected in the corners to shaft encoders. Thus, none of the coordinate generation is performed by the tablet or stylus, but rather through electro-mechanical networks connected to the stylus.

Function and Alphanumeric Keyboards

In addition to graphical interactions via the image, a program-fixed interaction is sometimes desired. A function keyboard merely provides buttons to which are assigned interpretations and do not require analysis of the image sequence. Again, the only connection is a logical one; indeed, the function keyboard does not require a display at all (whereas the tablet and light pen certainly have no function outside of the display environment). Perhaps the best explanation of the function keyboard is as follows:

A program has been written which allows drawing and scaling of the sketched image. The operator activates the light pen (or tablet), presses function keyboard button number 1 ("FKBI") to begin the program generation of a line - which he will sketch. When the pen is deactivated, the program retrieves the end coordinates and generates an image sequence for that line. To scale the line, the operator depresses FKB10, which scales up by a factor of 10. So on, until FKB32 is depressed, significantly the end of this display sequence (or session).

Another form of interactive device is the alphanumeric keyboard. Usually, this is an integral part of an alphanumeric display system. Under special instruction, a special character, called the "Cursor", is displayed on the face. The keyboard is similar to a standard typewriter keyboard; the cursor acts as the tracking cross does in pure graphics systems. On some systems, a key struck automatically inserts the character at that position and moves the cursor. On others, the buffer address of the character is read and, under program control, the new character inserted.

The alphanumeric keyboard may be used if alphabetic commands are desired. For example, the cursor is placed on the top of the screen. The operator types the request "LINE A(20,30), B(40,50)" and a line segment

labeled "A", "B" at either end is displayed from point (20,30) to point (40,50). A more conventional use of the alphanumeric keyboard is in editing of alphanumeric files, such as on-line bank balancing.

Miscellaneous Devices

The possibilities for interaction at a graphical display terminal are unlimited. Depending upon the application, one can be assured that new and novel types of devices will be developed, in parallel with new and sophisticated types of display systems. Some different types will be mentioned below.

- Toggle Switches: if a permanent sensing is desired, without the requirement of modifying the existing status of a function keyboard or complicated programming to use an alphanumeric keyboard, toggle switches are employed. The status of these switches can be sampled regularly by the display program.

- Analog Encoders: frequently, it is desired to use less abstract or rigid methods of communicating information than numbers or push buttons. In the perspective display systems (those enabling three-dimensional structures on the flat-screen), such as that at MIT's Project Mac, a "crystal ball" globe is used. The globe has three degrees of freedom in the upper hemisphere and shaft encoders determine the perspective desired by calculating, electrically, the three angles.

- Joy Stick; a joy stick is used very similarly to the crystal ball, to determine amount of displacement. It is conceptually very similar to the joy stick in aircraft.

PRINCIPLES OF DISPLAY DATA STRUCTURES

Donald P. Kelley

Research Assistant

Display Software

A trend developing in the area of interactive display programming is that of providing a programming language either as an extension of an existing language or as a dedicated language to facilitate graphic application programming. Because of the wide use of languages such as FORTRAN and the investment that has been made in such languages, an extension to include a graphical interface is desirable and feasible. However, the implementations of many of these languages may not be readily adaptable to the specification and processing of asynchronous interrupts which characterize interactive display processing; furthermore, these implementations are usually deficient in facilities to provide for dynamic programming and data management. An ideal language would have the widely known syntax and lexicon of ALGOL or FORTRAN, the list processing and re-entrant capabilities of LISP, COMIT, or SNOBOL and the time-shared interrupt processing of the TSS, CLII, and MCP systems.

The maintenance of the interrelationship between the components in a picture or drawing generally requires a more complex data structure than an alphanumeric file processing system which manages and displays a fixed character set. Line-drawing systems require an extensive set of routines for generation of conic sections; rotation, translation and scaling of parametric matrices; matrix clipping and magnification; point positioning; line-type control; and pen tracking. Alphanumeric systems usually consist of routines for text editing, message composing, program development, debugging, and generation of hierarchical or tree-like data structures.

Because of the similarity between alphanumeric display techniques and the techniques developed in the application of on-line real-time printers, only the programming techniques unique to displays which possess line generating characteristics will be considered in the sections which follow.

Commands

The vocabulary of an interactive computer graphic language is composed of imperative and interrogative commands. Although the syntax varies in

the various systems now in existance (4,5,9,10), the commands are generally similar. The commands of the TREET (11) list processing language used in the AESOP (10) language are: REPLACE, EVALUATE, KEYBOARD, STORE, ERASE, and ADD-RT. With the exception of ADD-RT, these commands are self-explanatory. ADD-RT means, "add the expression indicated by the first argument of the command to the file with the same parent as the node indicated by the light pen." The AESOP view action commands can be transmitted by light pen action or by on-line typewriter keyboard. The following is a list of alphanumeric commands:

RESTORE - This restores the display to its original position thereby cancelling all previous view actions.

BACKUP - Cancels the last previous view action.

KEYBOARD - Returns control to the on-line typewriter keyboard.

DEFTR - Redefines a function according to the present (modified) structure of the displayed tree.

TEST - Initiates the execution of the function being displayed.

The commands of GDL-2 (9) provide the imperatives necessary for a man-machine discourse dealing with two dimensional graphics. The commands PLOT and DISPLAY cause the initiation of line-drawing programs. PLOT activates x-y plotter routines and DISPLAY activates CRT display routines. The commands POINT, LINE, CIRCLE, ARC, DIST, and ANGLE call point and line generating routines. PRINT, ALL and FIGURE control the output of all or portions of files which describe line-drawings. Languages such as SKETCHPAD III (12) and GRASP (13) require even more specialized commands concerning the pitch, roll, and yaw of images in order to provide three-dimensional illusion.

Whether the above commands are typed, entered by light penning, or by pressing a command button, a signal must be generated to indicate to the processor that a new task has been assigned. Such signals are called interrupts.

Interrupt Handling

The topic of interrupt processing software is a part of the subject of executive and master control programs of operating systems but deserves consideration here because of its importance in effecting the dialogue of interactive display systems.

In an interactive display system the viewer is provided with a number of devices through which he can communicate with a computer program. These devices range from alphanumeric keyboards, function switches, and light pens (7) through voltage pens (8) and devices such as the RAND table (6). In addition, some display consoles are equipped with switches for the selection or rejection of data used to control the expansion or magnification of a displayed line drawing.

An alphanumeric keyboard usually causes one program interrupt for each symbol that is typed. When the digital representation of a symbol is available to the computer program, the program is interrupted in what it was doing in order to process the symbol. A light pen can be equipped with an activate switch which will cause a program interrupt or may be activated automatically by a given intensity of light. For example, in the BUIC Air Defense System the displayed data can be designated by the program as light-pen data. The light pen can be activated only when pointed at this particular class of data. A computer interrupt occurs when light-pen data designated by the light pen is reintensified on the display screen. The word or part of the word which caused the generation of that particular piece of data is then available to the computer program. The program can use this word in a match search of its data structure to determine what symbol or drawing was designated by the light pen.

Function switches have an associated activate button which when depressed will create a program interrupt. This interrupt then causes a word of information to be made available to the operating program. By examination of this word, the status of a function switch can be determined. More specifically, a bit of this word reflects the status of a certain

switch among the array of function switches. Interrupt processing permits any combination of light pen, keyboard and function switch operations.

Pen Tracking

When a coordinate point with no equivalent light target is desired, a light pen tracking routine is used. There are a number of methods which have proved successful as means of light pen tracking. Basically they accomplish tracking by determining the new position of the pen relative to its last position. The particular method for a given application is dictated by the display hardware and the amount of processing time available for the tracking function.

All methods of light pen tracking use a pattern of dots superimposed over a point known as the last position of the pen. One successful tracking pattern is the cross. To determine its current position, the last known position of the pen is used at the start point of a horizontal row of dots, first to the right and then to the left of the start point until they are not seen by the pen. Dots not seen by the pen do not cause an interrupt. The center point of this horizontal line is used as the new x-coordinate. Using this new x-coordinate vertical rows of dots are generated and a new center y-coordinate is similarly calculated, to locate the new pen-center position.

Time spent in displaying tracking crosses can be reduced by using a binary search to find the pen's outer edge. The first point is displayed at a point along the arm at ½ the radius of the pen's aperture, the second at half the distance of the first, etc., until the edge is found. Another similar method is to display four points on the circumference of the aperture of the last known point. If an edge point is inside the new pen aperture the next center point is some small increment in the same direction. If the calculated point falls outside the aperture the next center point is some small increment in the opposite direction.

Tracking crosses require that the pen not move further than the radius

of the aperture between samplings, otherwise, the center point is not seen and the pen is lost.

A spiral of dots around a previous pen position can be used to find or track a pen. When one of the dots is seen by the pen this becomes the new pen position. An error equal to the radius of the aperture is induced in this method.

Samples taken every five milliseconds allow a pen velocity of 25 inches per second for a pen aperture radius of 1/8 inch. A routine which operates in one millisecond therefore would require about 20 percent of computer time. For display systems not allowing this much time for tracking, prediction techniques must be used and data must be structured to minimize search time.

Display Data Structures

Experience up to this point has shown that data structured in conventional lists produced by structural languages such as LISP are not efficient or easy to use with multidimensional graphic data structure. The CORAL (4) (Class Oriented Ring Associated Language) language developed at the M.I.T. Lincoln Laboratory provides a general method of storing and manipulating arbitrarily complex information and a powerful language for describing and manipulating data forms. A data structure consisting of many elements pointing to each other and to appropriate measuring scales is called a plex (2) "an interconnected set of n-component elements." An n-component element is a single unit of information. Each of its components specifies one attribute or property of the element. Each component may be considered to be a pointer which addresses or describes a specific property.

An example of a very simple plex is a structure for specifying a line in two-dimensional coordinates as shown in Figure 1.

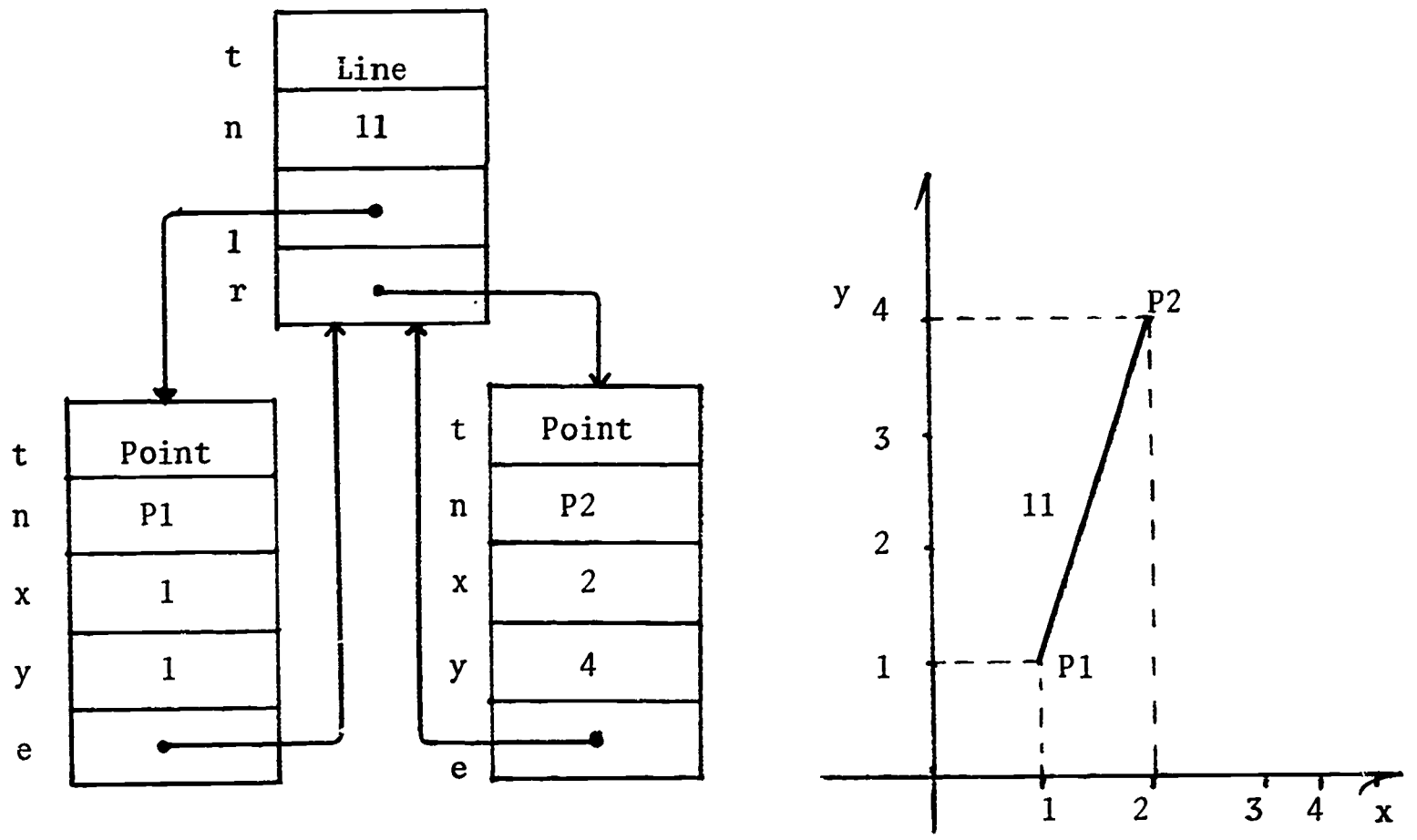


Figure 1: Plex for Specifying Line in 2-D Coordinates

Each line element contains a type component (t) designated LINE, a name component (n), and two additional components (l) and (r) which point to

its left and right end points. Both l and r point to elements of type POINT named P1 and P2 respectively, each have two additional components which specify the x- and y- coordinates of the point in some coordinate system and also a component (e) which indicates the endpoint relationship to the line l_1.

The basic n-component element structure can be expanded so that references made to a particular n-component element or block are collected together by a string of pointers which originates within the block. Starting from a line block an endpoint can be found. From the point block all lines which terminate on the point can be found by following the string of pointers which starts within the point block. The string of pointers closes with the last pointing back to the first to form a "ring." In addition the ring points both to the next and to the previous element making deletions as easy as insertions. To illustrate the usefulness of this type structure consider a triangle as described by the structure of Figure 2. In the trivial but illustrative problem of rotating the triangle θ degrees about point P1,

the point P2 is moved to;

$$x2' = x2 \cos \theta + y2 \cos \theta$$

$$x2' = x2 \sin \theta + y2 \cos \theta$$

and P3 is moved to;

$$x3' = x3 \cos \theta + y3 \cos \theta$$

$$y3' = x3 \sin \theta + y3 \cos \theta.$$

The coordinates (x2, y2) and (x3, y3) for the triangle T1 are readily found and modified, and no other modifications to the describing structure is required to define the triangle T1.

For an application of this structure to the analysis of electronic circuits see Evans and Katzenelson.

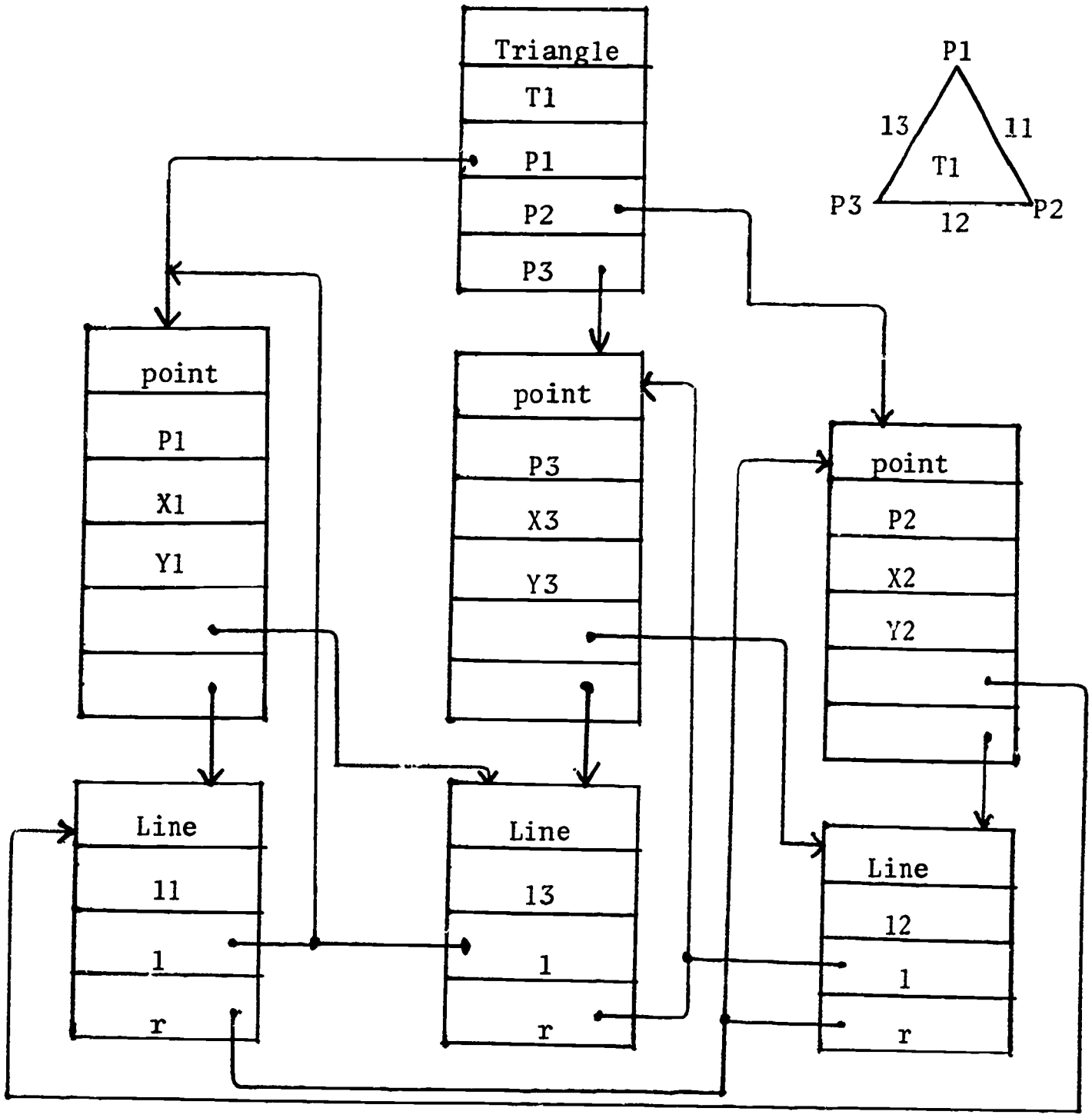


Figure 2: Plex with Ring Structure

The need of the ring structure becomes apparent in the drawing of figures which are composed of a number of previously drawn figures. The effectiveness of an interactive display system can be measured by the extent that its software provides for subpicture constructs.

Subpicture Structure

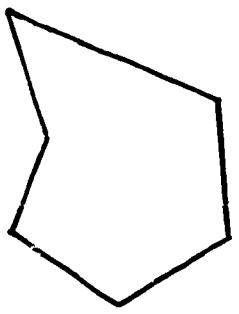
The subpicture (5) capabilities of an interactive display system allows the generation of figures of any complexity which are composed of huge numbers of similar parts. A drawing may be reduced in size and reproduced anywhere with help of the light pen. Subpictures can be recursive within subpictures, attached to subpictures or used singly with other simple symbols.

Figure 3 illustrates a construction using subpictures. The group of seven hexagons can be treated as a single symbol. The data structure defining a symbol can be readily stored and recalled if needed in another construct at a later time.

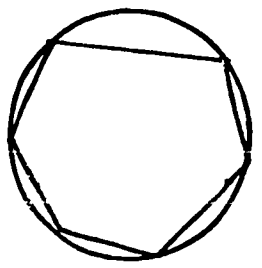
Graphic Processing

The common method of specifying geometric figures is in implicit form, i.e., $F(x,y,) = 0$. The equation in implicit form of a straight line through the points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ is $x_1(y - y_2) - y_1(x - x_2) = 0$. However the implicit form does not readily lend itself to display programming because 1) it describes the entire figure, not a point of the figure which is what is displayed and 2) except for the simpler two dimensional equations the variables cannot be easily separated (for example, into an equation of the form $y = f(x)$). Therefore parametric equations (in which each variable can be represented independently) are generally used. The equations to represent the line using a parameter t where t may be thought of as the time taken to move from one point on the line to another are $x = f(t) = (x_2 - x_1)t + x_1$ and $y = g(t) = (y_2 - y_1)t + y_1$. In particular the equations represent the motion of a "particle" along a curve where at any instant t the coordinates of the "particle" are $x = f(t)$ and $y = g(t)$. The density of the "particles" necessary to produce the desired curve determines the number of discrete values of t where $0 \leq t \leq 1$.

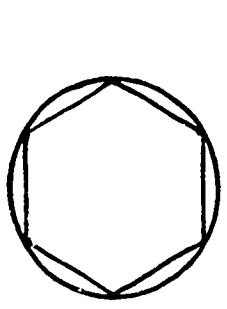
The coordinates of a symbol or point once determined must be converted to coordinates so that the symbol or point can be displayed. Usually this



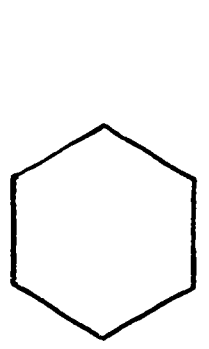
a) A six sided figure



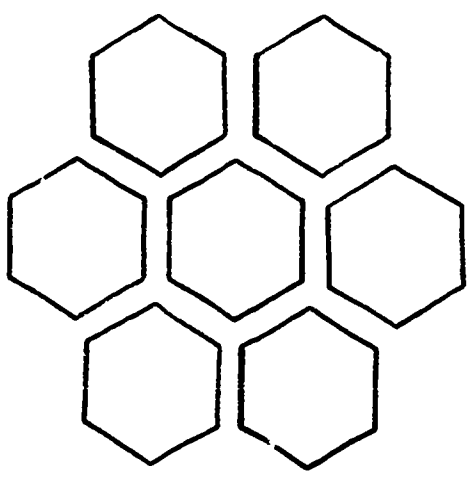
b) is inscribed in a circle by moving each corner onto the circle.



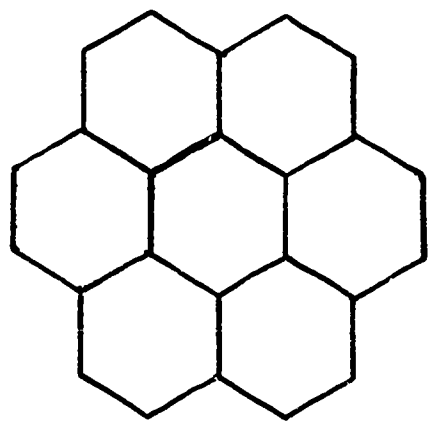
c) The sides are made equal,



d) the circle is erased, and each corner is designated as an attachment point.



e) Seven hexagons are recalled and



f) joined at corners.

involves determining the area to be displayed and converting all that falls within the area from "paper coordinates" to scope coordinates. This is in a sense a "view box" which represents in "paper coordinates" the area to be displayed scaled to the display area. By providing a displayed "view box" which upon request by the viewer can be adjusted in size and position a particular portion of the display can be magnified or demagnified by adjusting the "paper coordinates" of all that falls within the box.

This discussion has not considered the interesting, complex area of 3-dimension display programming. Much has been done in this area (12), but the problem of invisible lines (lines that should not be shown in a 3-D display) has reduced its effectiveness. Current work indicates that animated 3-D figures will soon appear on the VDT.

In conclusion, the questions which must be answered when designing a communication program for viewer machine interface are:

1. What is the organization of the data related to the mission and the display?
2. By what means will display of program be referenced?
3. What is a systematic way of providing algorithms which use the available data to perform the functions required by the system?

REFERENCES

1. Roberts, L. G., "Graphical Communication and Control Languages" Information System Sciences, Spartan Books, Washington, D.C.

2. Ross, D. T., and Rodriquez, J. E., "Theoretical foundations for the computer-aided design system," 1963 Proc. SJCC, vol. 23, pp. 305-322.

3. Evans, D. S. and Katzenelson, J., "Data structure and man-machine communication for network problems," Proc. IEEE, vol. 55, pp. 1135-1144, July 1967.

4. Sutherland, W. R., "The CORAL language and data structure," M.I.T. Lincoln Lab., Lexington, Mass., Tech Rept. 405.

5. Sutherland, W. R., "Sketchpad: a man-machine graphical communication system," M.I.T. Lincoln Lab., Lexington, Mass., Tech Rept. 296, January 30, 1963.

6. David, M. R. and Ellis, T. O., "The RAND tablet: a man-machine graphical communication device," 1964 Proc. FJCC, vol. 26, pp. 325-331.

7. Stotz, R., "Man-machine console facilities for computer-aided design," 1963 Proc. SJCC, vol. 23, pp. 323-328.

8. Jacks, E. L., "A laboratory for the study of graphical man-machine communications," 1964 Proc. FJCC, vol. 26, pp. 343-350.

9. Henderson, D. A., Jr. "A graphics display language," Department of Computer Science, University of Illinois, Urbana, Illinois, Rept. No. 240, August 21, 1967.

10. Bennett, Edward, Haines, E. C., and Summers, J. K., "AESOP: a prototype for on-line user control of organizational data storage, retrieval and processing," 1965 Proc. FJCC vol. 27 part 1 pp. 435-455.

11. Haines, E. C., "The TREET list processing language." SR-133, The MITRE Corporation, April 1965.

12. Johnson, T. E., "Sketchpad III: three dimensional graphical communication with a digital computer," M.I.T. Electronics Systems Laboratory, Rept. ESL-TM-173, May 1963.

13. Thomas, E. M., "GRASP: a graphic service program." Proc. 22nd National ACM Conference, 1967, pp. 395-402.